CSE 574 - Introduction to Machine Learning - Programming Assignment 1

Classification and Regression

Group 10

Akshaya Krishnan - ak243
Dithya Sridharan - dithyasr
Vijay Jagannathan - vijayjag

Problem 1 : Experiment with Gaussian Discriminators

Problem 1 asks us to implement :-

- **Linear Discriminant Analysis** (LDA)
- **Quadratic Discriminant Analysis** (QDA)

This can be done by implementing the following functions :-

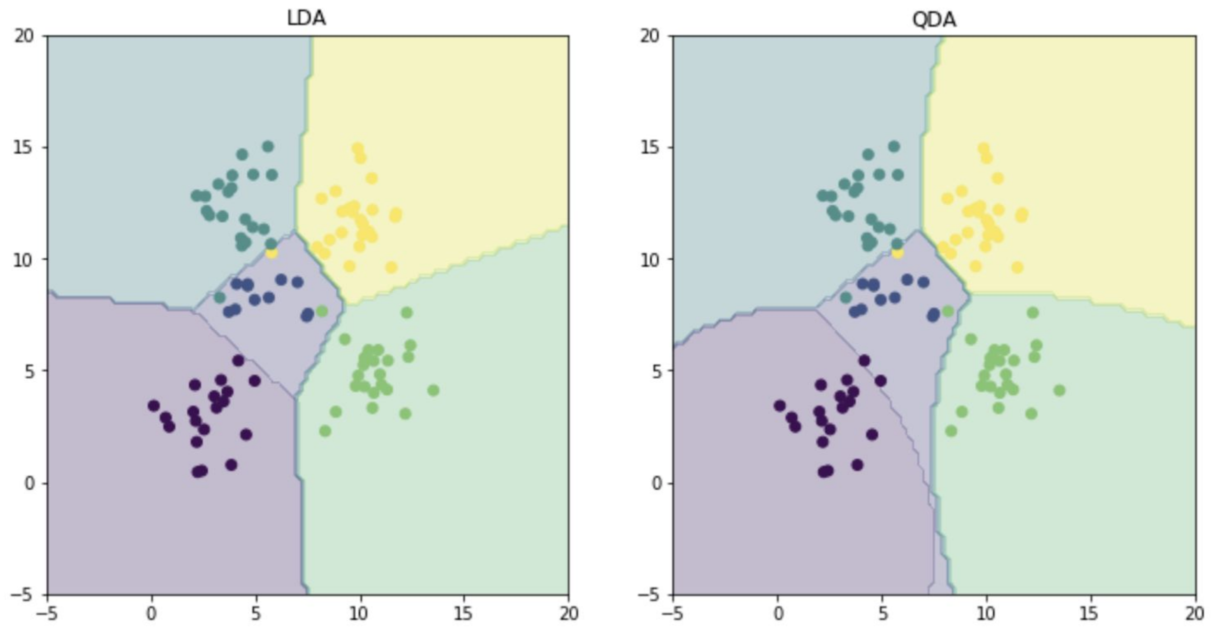
1. **ldalearn**() - takes a training data set as input (sample_train) and returns the corresponding means and covariance matrix.
2. **qdalearn**() - takes a training data set as input (sample_train) and returns the corresponding means and covariance matrix.
3. **ldatest**() - returns the true labels for some test data (sample_test) and also returns the accuracy using the true labels for this test data.
4. **qdatest**() - returns the true labels for some test data (sample_test) and also returns the accuracy using the true labels for this test data.

In general, LDA and QDA are aimed at performing classification of data into classes. The LDA classifier assumes that each class comes from a normal distribution with a class-specific mean vector and a common variance. We utilize LDA to estimate the parameters so that we can leverage the Bayes classifier. LDA has different means but the same variance. For calculating variance, LDA uses Mahalanobis distance to calculate the class. QDA also uses mean and variance but is different from LDA because it uses non-linear combination of variables to predict the classes.

On implementing the above functions, we found out that:-

- **LDA** accuracy is **97.0**
- **QDA** accuracy is **94.0**

<u>LDA and QDA Plot</u>



- LDA uses Mahalanobis distance, which is an efficient method to calculate variance. QDA is similar to LDA but it has a separate covariance for each class. Since QDA assumes a quadratic decision boundary, it can accurately model a wider range of problems than the linear methods. But QDA will suffer when it is a high variance data set. Our dataset is relatively small. QDA and LDA have given good accuracy.
- Since QDA uses different covariance for each class the categorical boundary changes. This is why LDA gives a linear boundary while QDA gives a non-linear boundary.

Problem 2: Experiment with Linear Regression

This can be done by implementing the following functions :-

1. **learnOLERegression**() - takes X_{train} and y_{train} as input, where, X_{train} is the input data matrix and y_{train} is the target vector and returns the weight vector(w).
    - Implement ordinary least squares method to estimate regression parameters by minimizing the regularized squared loss.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}$$

    - Weight vector 'w'

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

2. **testOLERegression**() - takes X_{test}, y_{test} and w as input, where, X_{test} is the input data matrix, y_{test} is the target vector and w is the weight vector to find the Mean Squared Error (MSE).
    - Find the Mean Squared Error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

In general, Linear regression attempts to model the relationship between two variables by fitting a linear equation to the observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. The weights that maximize the training data is calculated and the regression model is built. Prediction is done for two cases :

1) With Intercept
2) Without Intercept

Mean Squared Error values for training and test data :-

1. **Training Data**
    - MSE without intercept - **19099.44684457**
    - MSE with intercept - **2187.16029493**
2. **Test Data**
    - MSE without intercept - **106775.36155223**
    - MSE with intercept - **3707.84018148**

When we don’t include the constant, the line is forced to go through the origin. When the regression model fitting line doesn’t go through the origin, regression coefficients and predictions will be biased. The inclusion of the intercept does not force the model to go through the origin, hence MSE with intercept is lesser for training and test data. Therefore, **MSE with intercept is better for both training and test data**.

Problem 3: Experiment with Ridge Regression

Problem 3 asks us to implement:-

- Parameter estimation for ridge regression by minimizing the regularized squared loss.

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}$$

  This squared loss can also be written in matrix notation as:-

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{1}{2}\lambda \mathbf{w}^\top \mathbf{w}$$

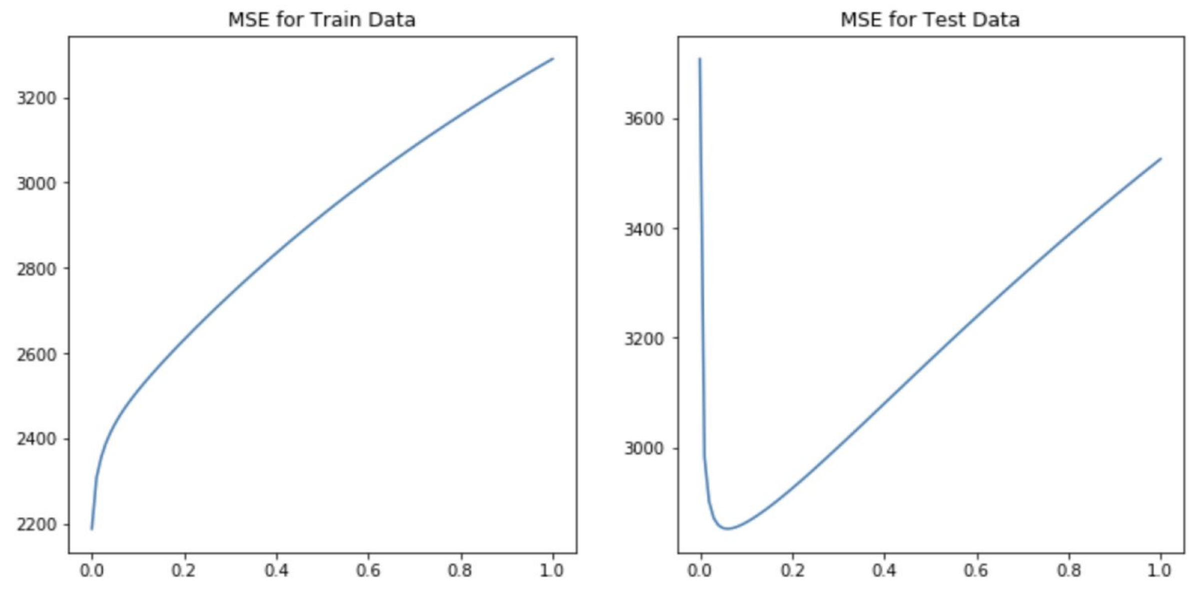
This can be done by implementing the following functions:-

1. **learnRidgeRegression**() - takes X_{train} and y_{train} as input, where, X_{train} is the input data matrix and y_{train} is the target data vector

Mean Squared Error for training and test data :-

1. Training Data
   - MSE with Intercept - **2433.1744367**
2. Test Data
   - MSE with Intercept - **2852.66573517**

MSE for Training and Test Data - Plot



Comparison between Linear and Ridge Regression in terms of MSE on training and test data :-

1. **Linear Regression**
   - Training Data - MSE with Intercept - **2187.16029493**
   - Test Data - MSE with Intercept - **3707.84018148**
2. **Ridge Regression**
   - Training Data - MSE with ($\boldsymbol{\lambda}$ = 0.05) - **2433.1744367**
   - Test Data - MSE with ($\boldsymbol{\lambda}$ = 0.05) - **2852.66573517**

In general, Ridge regression is a technique for analyzing multiple regression data that suffer from multicollinearity. Ridge Regression is like least squares but shrinks the estimated coefficients towards zero. From the weights obtained, by adding some bias to the ridge regression estimates, we are able to reduce the standard errors. Higher values of lambda produce less flexible functions and higher training errors. Lower values of lambda produce the most flexible functions and these will fit the training data more accurately and therefore produce lower training error rates. The **optimal value of lambda is the value that produces the lowest test error rates**. This value is found when the test error is 2852 and therefore the **optimal lambda value is 0.05**.

Problem 4: Using Gradient descent for Ridge Regression Learning

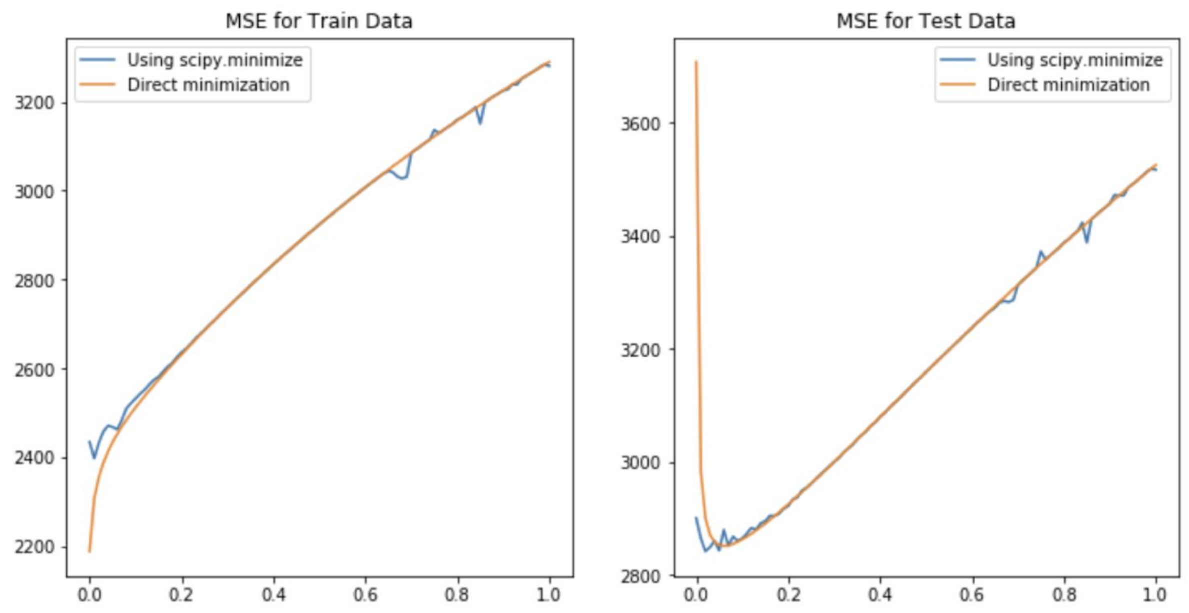
Problem 4 asks us to :-

- calculate the regression parameters directly without using any analytical expressions.
- We can use gradient descent to minimize the loss function and estimate the weights **w**.

This can be done by implementing the following functions:-

1. **regressionObjVal**() - to compute the regularized squared error and its gradient with respect to **w**.

MSE for Training and Test Data using scipy.minimize and direct minimization



Comparison between Ridge Regression with and without gradient descent in terms of MSE on training and test data :-

1. **Ridge Regression without Gradient Descent**
    - Training Data - MSE with (λ = 0.05) - **2433.1744367**
    - Test Data - MSE with (λ = 0.05) - **2852.66573517**
2. **Ridge Regression with Gradient Descent**
    - Training Data - MSE with (λ = 0.05) - **2467.3097612**

- Test Data - MSE with (λ = 0.05) - **2843.08348391**

Since matrix inversion is expensive, we use Gradient descent to avoid such inverse calculations making the model fast and efficient. In the graphs, we are comparing MSE obtained with and without using scipy minimize function. The blue line indicates MSE with minimize function and the orange line indicates MSE without using minimize function. The blue line has some dips and shoots, for some lambda values.

Why gradient descent gives better results?
When we implement the gradient descent method, we include the lambda term. This gives us more control on determining how low we can go in order to obtain the local minima. Thus, gradient descent helps us to find the unique minimum value.

Problem 5: Non-linear Regression

Problem 5 asks us to:-

- Investigate the impact of using higher order polynomials for the input features.
- For this, we use only the third variable as the input variable

```
x_train = x_train[:,2]
x_test = x_test[:,2]
```

This can be done by implementing the following functions:-

1. **mapNonLinear**() - takes a single attribute x and converts it into a vector of p attributes as shown below

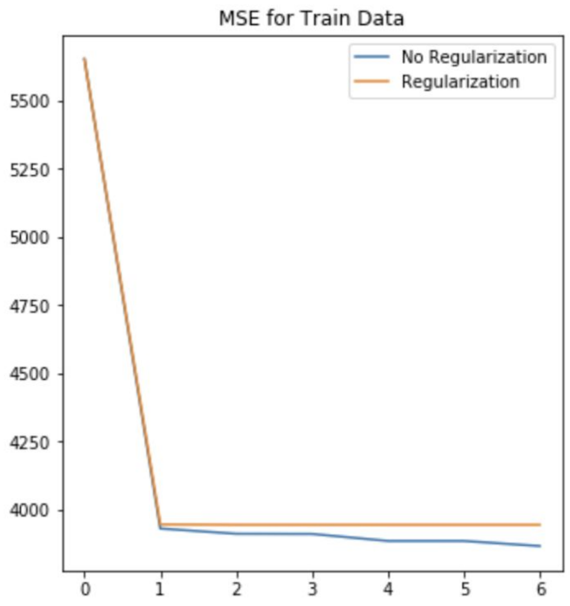
$$1, x, x^2, \ldots, x^p$$

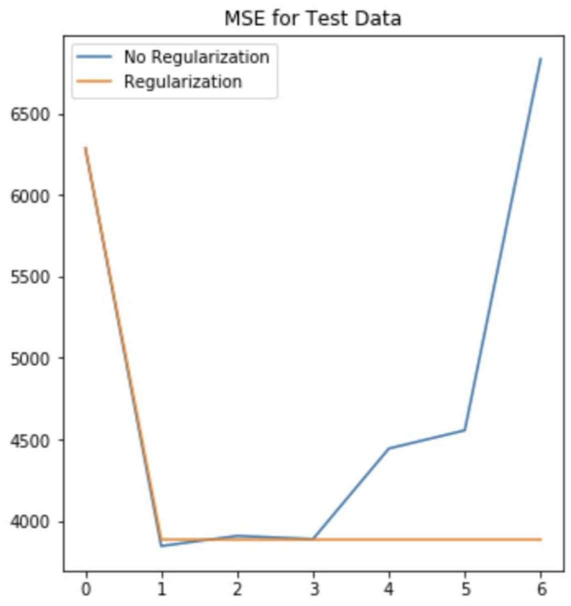
On comparing the MSE values for Training and Test Data for different lambda and p values, we get the following results:-

p	**Training Data(λ = 0)**	**Test Data(λ = 0)**	**Training Data(λ = 0.05)**	**Test Data(λ = 0.05)**
0	5650.710538897617	6286.404791680897	5650.711489069524	6286.802264172339
1	3930.915407315901	3845.0347301734146	3945.9948342861508	3884.695622903158
2	3911.839671204956	3907.1280991079366	3944.6776809012486	3884.545679026596
3	3911.18866493145	3887.975538236015	3944.673052639404	3884.546641924834
4	3885.4730681122714	4443.327891813341	3944.6728330323867	3884.546635661096
5	3885.4071573970805	4554.830377434614	3944.672831291066	3884.546636896555
6	3866.8834494460493	6833.459148719649	3944.672831252616	3884.546636929001

- For the first power of input, the error value drops for with and without regularization. Hence, **optimal p value is 1**.

MSE for both Training and Test Data(with and without Regularization)





- In this problem, we are passing (**λ = 0**) for non-regularization. To include the regularization term, we are passing (**λ = 0.05**, optimum lambda value from problem 3) as a parameter to the function.

- We plot the graph for both train and test error with and without regularization. We observe that, for the case with regularization, the MSE value drops at (**p = 1**) and then the trend seems to continue. But, without regularization, the MSE value increases suddenly at (**p = 3**). So, we should include the regularization term.

Problem 6: Interpreting Results

In the above problems, we have implemented 5 models to predict diabetes levels.

Model Type	Training Data Error	Test Data Error
Linear Regression without Intercept	19099.44684457	106775.36155223
Linear Regression with Intercept	2187.16029493	3707.84018148
Ridge Regression without Gradient Descent (λ = 0.05)	2433.1744367	2852.66573517
Ridge Regression with Gradient Descent (λ = 0.05)	2467.3097612	2843.08348391
Non-Linear Regression(λ = 0.05, p = 1)	3945.994834286150	3884.6956229031584

On comparing the results from the previous problems, we can see that **Ridge Regression using gradient descent** gives the least test error

The metrics used to choose the best model:

- Least test error
- Optimal lambda value

So we would recommend ridge regression using gradient descent model to predict diabetes value.